

HOWTO

Build and run on POSIX-compatible systems

依赖（假设图形界面已安装）：make, gcc, gtk+-3.0, libao, libmpg123, pkg-config;

Ubuntu:

```
$ sudo apt install build-essential libao-dev libmpg123-dev libgtk-3-dev
```

命令：make build_all; make run;

Build and run on Windows

环境：msys2 (暂时没能打包成功，还是需要安装运行时);

依赖：make, gcc, gtk+-3.0, pkg-config, libao, libmad;

```
$ pacman -S mingw64/mingw-w64-x86_64-gcc make mingw64/mingw-w64-x86_64-pkgconf
```

```
$ pacman -S mingw64/mingw-w64-x86_64-libao mingw64/mingw-w64-x86_64-libmad
```

```
$ pacman -S mingw64/mingw-w64-x86_64-gtk3
```

命令：make build_win; make run_win;

验收

Demo1.mp4: 多次高速电脑自动操作和一次常速完整的电脑自动操作;

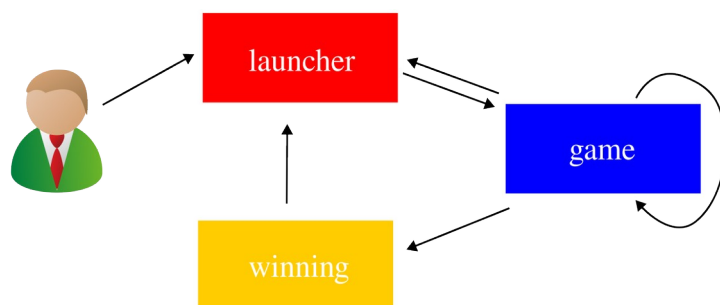
Demo2.mp4: 项目设计介绍;

Caterpillar FlightChess V3 项目汇报

一、架构

1. 程序模块

程序分为三个模块：launcher, game 和 winning。



图示 1：程序模块

其中，用户启动 launcher，launcher 检查并根据用户需要修改配置文件，然后启动 game，进入正式的游戏环节。game 结束后启动 winning，winning 宣布获胜者，被用户关闭后回到 launcher。在 game 中也可以回到 launcher 或者重启 game。

2. 设计模式

面向对象的设计模式，模块化的代码组织方式。

<pre>Place int id; //通过其父类代码在 game 中注册 int x, y; //coordinates in GUI Place* adjacence; vector<Place*> exit; vector<Place*> special; exit; set<Chess*> chesses; set.coordinate(int x, int y); void set.id(int id); void set.exit(vector<Place*> exit); void set.adjacence(Place* adjacence); void get.adjacence(Place* adjacence); void get.adjacence(Chess* chess); Place* get.special.exit(Chess* chess); Place* get.exit(Chess* chess); Place* add.chess(Chess* chess); void remove.chess(Chess* chess); void //移除 chess 对象时，用 Player::take hold</pre>	<pre>Chess Player* player; Place* place; string color; //chess image filename GtkWidget* widget; move.to(Place* place); void name(int step); void ! conflict(int step); bool set.place(Place* place); void set.color(string color); void gui.show(); void gui.follow.dest(int step); void</pre>	<pre>Player int id; //用户号... int number_of_chess; int type; string color; Place* airport; vector<Chess*> chesses; //保存所有棋子坐标 Place* airport; set.id(int id); void init.chess(int n); void set.airport(Place* airport); void set.color(string color); void set.type(int type); void action(); void //处理下棋逻辑 take.back(Chess* chess); void //把 chess 放回 airport</pre>	<pre>GameInfo int id; //游戏号... vector<Player*> players; vector<Place*> airport; int number_of_players; int number_of_places; bool control_mode; Place* exit; GtkWidget* dice_label; GtkWidget* message_list; GtkWidget* board; void read_info(stream& in); //从流中读取信息 void set_player_type(int type); int dice(); int gui_dice(); int roll(); //根据 control_mode //根据骰子决定 //是否 gui_dice void gui_roll_string(); //更新 gui message_list int main(); //主函数</pre>
--	--	--	--

图示 2：对象

面向对象设计主要体现在 game 模块中，游戏的主要数据结构是 Place, Chess, Player。棋盘的链表表示法灵感来自于隔壁 SICP proj03。

模块化的代码组织方式主要体现在音频模块。mp3player.hpp 中的 Mp3Player 类封装了用 libao 和 libmpg123 库实现的简单的音乐播放器，在三个程序模块中都有使用。

二、基础功能

1. 设置玩家数和玩家类型：在 launcher 模块中设置；
2. 显示棋盘（经典版和作业版）：在 game 模块中显示，棋盘动态加载；
3. 防外挂骰子：ctpchessV3.cpp::dice()和 ctpchessV3.cpp::gui_dice()函数确保正常运行时点数为 1~6，adt.hpp::intPrompt()函数确保 DEBUG 模式时点数仍是 1~6；
4. 基础规则：非 6 不出机场，终点反弹，同格子情况处理，电脑玩家自动操作，游戏胜利，……；

三、拓展功能

1. 飞、跳：通过链表型棋盘和自定义的地图格式实现；
2. 动态加载的地图。程序从文件中动态载入棋盘的描述、背景、棋子的样式等，使能够在不修改程序的条件增加新地图、设计新规则，可以载入作业要求的地图和原版地图，也可以后续加入新地图。设计的地图格式参见 map/ctpchessmap-def.conf。
3. 图形界面。如图所示，用 gtk 实现了简陋的图形界面。



图示 3：GUI



4. 背景音乐。如演示视频所示，实现了简单的背景音乐播放。
5. 人工智障。时间原因，只实现了一个简单的策略。（`ctpchessV3.cpp::Player::action()`）

四、完成项目中解决的问题/学到的知识

1. 多线程线程同步和图形界面串行访问的问题，同步/异步处理

在老师和助教的帮助下，学习了 `std::mutex` 和 `semaphore.h`，并实现了符合程序逻辑的 `C_SIMPLE_SIGNAL` 处理函数族。

2. 图形库 GTK，MP3 解码库 libmpg123，音频输出库 libao 的使用，PCM 编码的基础处理

在学习库的使用过程中，体会到软件开发全流程中各种坑和各种“技术”的真实含义。

... ..

五、程序的优点

1. 通用性。程序设计的结构使得该程序能够加载各类地图，甚至稍加修改，就能变成地图类型相似的游戏，如植物大战僵尸（SICP proj03 Ants VS Somebees 😊）。
2. 可移植性。程序使用了 STL，GTK，libmpg123，libao 等跨平台可移植的库，使得程序本身可移植。
3. 代码易于理解。

六、程序的已知缺陷、不足

1. 终点的设计。程序中终点处棋子相互堆叠，只显示最后加入棋盘的棋子，不知道对于每一个玩家有几个棋子到达。
2. 因为先实现了主要逻辑，后加了图形界面，然而 GTK 的要求是所有图形界面操作在主 thread 中完成，而我的程序在另一个线程运行，需要将后续的操作串行化，导致图形界面部分的代码不太优雅，有时打破抽象。

七、总结

第一次写小型项目，总体方向是对的，先设计架构，定义模块，然后定义接口，最后具体实现。但是设计架构时过于自信，目标太高，而具体实现中又有许多细节问题需要处理，踩了不少坑，但也收获了相应的经验，是一次有益的尝试。

八、COPYRIGHT

GPL v2。详见 COPYING。程序使用的库许可如下：

gtk+-3.0: LGPL v2.1

libmpg123: GPL v2

libao: LGPL v2